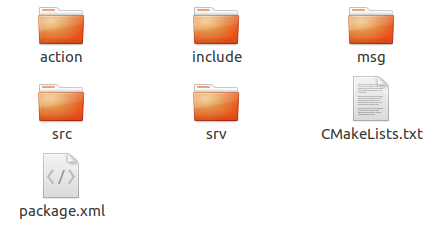
# ROS之动作编程

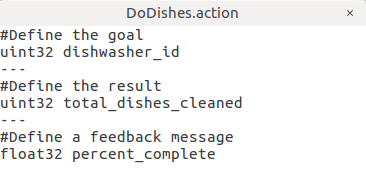
本实验中，客户端向服务端发起一个目标，服务端启动目标，并在完成目标过程中，以一定频率返回完成任务的状态，直到目标完成后，实现如下。

1. 自定义Action

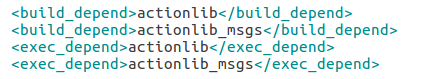
在功能包下新建一个action文件夹



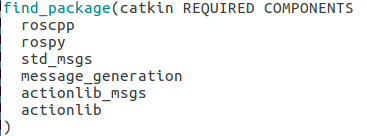
创建一个DoDishes.action文件，内容如下

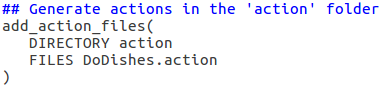


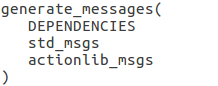
1. 在package.xml中添加功能包依赖：



1. 在CMakeList.txt中添加：







1. 回到空间根目录，尝试编译，确保没问题。
2. 编写客户端程序和服务端程序：

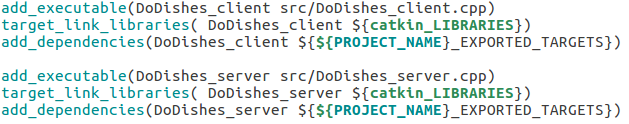
DoDishes\_server.cpp

|  |
| --- |
| #include <ros/ros.h>  #include <actionlib/server/simple\_action\_server.h>  #include "learning\_communication/DoDishesAction.h"  **typedef** actionlib**::**SimpleActionServer**<**learning\_communication**::**DoDishesAction**>** Server**;**  void execute**(**const learning\_communication**::**DoDishesGoalConstPtr**&** goal**,** Server**\*** as**)**  **{**  ros**::**Rate r**(**1**);**  learning\_communication**::**DoDishesFeedback feedback**;**  ROS\_INFO**(**"Dishwasher %d is working."**,** goal**->**dishwasher\_id**);**  **for(**int i **=** 1**;** i**<**10**;** i**++)**  **{**  feedback**.**percent\_complete **=** i **\*** 10**;**  as**->**publishFeedback**(**feedback**);**  r**.**sleep**();**  **}**  ROS\_INFO**(**"Dishwasher %d finish working."**,** goal**->**dishwasher\_id**);**  as**->**setSucceeded**();**  **}**  int main**(**int argc**,** char **\*\***argv**)**  **{**  ros**::**init**(**argc**,** argv**,** "do\_dishes\_server"**);**  ros**::**NodeHandle n**;**  Server server**(**n**,** "do\_dishes"**,** boost**::**bind**(&**execute**,** \_1**,** **&**server**),** **false);**  server**.**start**();**  ros**::**spin**();**  **return** 0**;**  **}** |

DoDishes\_client.cpp

|  |
| --- |
| #include "actionlib/client/simple\_action\_client.h"  #include "learning\_communication/DoDishesAction.h"  **typedef** actionlib**::**SimpleActionClient**<**learning\_communication**::**DoDishesAction**>** Client**;**  void doneCb**(**const actionlib**::**SimpleClientGoalState**&** state**,** const learning\_communication**::**DoDishesResultConstPtr**&** result**)**  **{**  ROS\_INFO**(**"Yay!, The dishes are now clean"**);**  ros**::**shutdown**();**  **}**  void activeCb**()**  **{**  ROS\_INFO**(**"Goal just went active"**);**  **}**  void feedbackCb**(**const learning\_communication**::**DoDishesFeedbackConstPtr**&** feedback**)**  **{**  ROS\_INFO**(**"Percent\_complete: %f "**,** feedback**->**percent\_complete**);**  **}**  int main**(**int argc**,** char**\*\*** argv**)**  **{**  ros**::**init**(**argc**,** argv**,** "do\_dishes\_client"**);**  Client client**(**"do\_dishes"**,** **true);**  ROS\_INFO**(**"Waiting for action server to start."**);**  client**.**waitForServer**();**  ROS\_INFO**(**"Action server started, sending goal."**);**  learning\_communication**::**DoDishesGoal goal**;**  goal**.**dishwasher\_id **=** 1**;**  client**.**sendGoal**(**goal**,** **&**doneCb**,** **&**activeCb**,** **&**feedbackCb**);**  ros**::**spin**();**  **return** 0**;**  **}** |

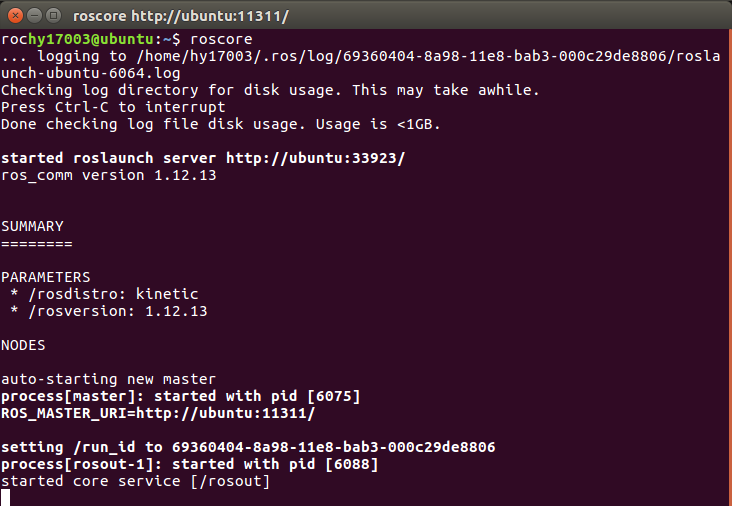
1. 在CMakelList.txt中添加：



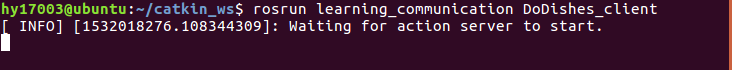
1. 编译

8．测试

打开一个终端，启动rowcore



打开一个终端，启动客户端：



打开一个终端，启动服务端：



此时，客户端同时输出：

